An efficient multi-objective genetic algorithm for cloud computing: NSGA-G

Trung-Dung Le
Univ Rennes
CNRS, IRISA
Lannion, France
trung-dung.le@irisa.fr

Verena Kantere
University of Ottawa
School of Electrical Engineering and Computer Science
Ottawa, Canada
vkantere@uOttawa.ca

Laurent d′Orazio
Univ Rennes
CNRS, IRISA
Lannion, France
laurent.dorazio@irisa.fr

***Abstract*—Cloud computing provides computing resources with elasticity following a pay-as-you-go model. This raises Multi-Objective Optimization Problems (MOOP), in particular to find Query Execution Plans (QEPs) with respect to users' preferences being for example response time, money, quality, etc. In such a context, MOOP may generate Pareto-optimal front with high complexity. Pareto-dominated based Multi-objective Evolutionary Algorithms (MOEA) are often used as an alternative solution, like Non-dominated Sorting Genetic Algorithms (NSGAs) that provide better computational complexity. This paper presents NSGA-G, a NSGA based on Grid Partitioning for improving complexity and quality of current NSGAs. Experiments on DTLZ test problems using Generational Distance (GD), Inverted Generational Distance (IGD) and Maximum Pareto Front Error prove the relevance of our solution.**

***Index Terms*—Multi-objective Optimization, Pareto-optimal solutions, Genetic algorithms, Non-dominated Sorting Genetic Algorithm**

I. Introduction

Cloud computing provides computing resources with elasticity following a pay-as-you-go model. This raises Multi-Objective Optimization Problems (MOOP), in particular to find Query Execution Plans (QEPs). Indeed, Users may face conflicting objectives. Let's consider a query **Q** in a example below.

Example 1.1: A query **Q** in the medical domain, based on TPC-H query 3 and 4 [1] be:

```
SELECT p.UID, p.PatientID, s.PatientName,
       p.PatientBrithDate, p.PatientSex,
       p.EthnicGroup, p.SmokingStatus,
       s.PatientAge, s.PatientWeight,
       s.PatientSize, i.GeneralName,
       i.GeneralValues, q.UID,
       q.SequenceTags, q.SequenceVRs,
       q.SequenceNames, q.SequenceValues
FROM Patient p, GeneralInfoTable i,
       Study s, SequenceAttributes q
WHERE p.UID = s.UID
       AND p.UID = i.UID
       AND p.UID = q.UID
       AND p.PatientSex = 'M'
       AND p.SmokingStatus = 'NO'
       AND s.PatientAge >= x
       AND q.SequenceNames
       LIKE '%X-ray%'
```

This query is optimized and transformed into a logical plan. After that a physical tree is generated and then translated into physical tasks with respect to the infrastructure and its configuration. From a physical tree, various QEPs are generated with respect to the number of nodes, their capacity in terms of CPU, memory and disk and the pricing model. Table I presents an example of possible QEPs for **Q**. Choosing an execution plan is a tradeoff between objectives such as the response time or the monetary cost, and depends on users' preferences: a user A may prefer minimizing his budget (QEP1); a user B may want the lowest response time (QEP2); a user C may look for a tradeoff between time and money (QEP3).

TABLE I: Multiple Objectives for Query Execution Plans

QEP	Vms	Price ($\$/60min$)	Time (min)	Monetary ($)
QEP1	10	0.02	60	0.2
QEP2	40	0.02	22	0.29
QEP3	30	0.02	26	0.26

MOOP can be defined as [2]:

$$minimize(F(x) = (f_1(x), f_2(x)..., f_m(x))^T)$$

where $x = (x_1, ..., x_n)^T \in \Omega \subseteq \mathbb{R}^n$ is an n-dimensional vector of the decision variables, Ω is the decision (variable) space and F is the objective vector function that contains m real value functions. In general, there is no point in Ω that minimizes all the objectives together. Pareto optimality is defined by tradeoffs among the objectives. If there is no point $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$, $x^* \in \Omega$, x^* is called Pareto optimal, and $F(x^*)$ is called a Pareto optimal vector. A set of all the Pareto optimal points is the Pareto set. A Pareto front is a set of all the Pareto optimal objective vectors.

Generating the Pareto-optimal front can be computationally expensive and is often infeasible [3]. In the context of cloud computing, a challenging problem is thus: how to efficiently propose diverse QEPs?

As an alternative to the Pareto-optimal, Evolutionary Algorithms (EAs) look for approximations (set of solutions not far away from the optimal front). Evolutionary Multi-objective Optimization (EMO) approaches [4]–[8] have been developed

based on Pareto dominance techniques [4], Pareto Archived Evolution Strategy (PAES) [5], Strength Pareto Evolutionary Algorithm (SPEA) [6]. EMO algorithms aims to find and maintain a front of non-dominated solutions, to find optimal Pareto set solutions. An evolutionary process enables to explore the search space. The non-dominated set is maintained and prepared for the next population of candidate solutions.

Among EMO approaches, Non-dominated Sorting Algorithms (NSGAs) [7], [9] aims to reduce computational complexity while maintaining diversity among solutions. NSGA-II [7] and SPEA-II [6] use crowding distances to maintain diversity. However, complexity is high and diversity cannot be preserved with more than two objectives [10]. Zhang and Li proposed an algorithm maintaining diversity with more than three objectives problem, MOEA/D [2]. MOEA/D uses a decomposition approach to divide multiple objectives into various single objective optimization subproblems and solved up to four objectives [11]. In 2013, Deb and Jain proposed NSGA-III [8], using a set of reference directions to guide the search process. However the computation complexity is the highest among algorithms.

This paper presents NSGA-G (Non-dominated Sorting Algorithm based on Grid Partitioning) to improve both diversity and efficiency. NSGA-G maintains diversity by selecting solutions in a Pareto set in multiple groups randomly, which are divided by a Grid Partitioning in the space of solutions. A solution is selected by comparison in a group, instead of between all members in a Pareto set. This approach reduces the computation time in EMO with large populations. NSGA-G does not only inheritances of the superior characteristics of NSGAs in computational complexity, but also improve both quality and computation time to solving MOOP in cloud computing becoming easier. NSGA-G has been validated though experiments on DTLZ problems [12].

The remainder of this paper is organized as follows. Section 2 presents the background of our research. NSGA-G is presented in Section 3. Section 4 presents experiments to validate NSGA-G. Finally, Section 5 concludes this paper and lists some perspectives.

II. Backgroud

As a running example let a query **Q** in *Example 1.1*. Let's assume the query is processed on Amazon EC2. The master consists in a m2.4xlarge instance (8 virtual cores and with 68.4 GB of RAM). Workers consist in m3.2xlarge instances (8 virtual cores and with 30 GB of RAM). If the pool of resources is 70 VCPU with 260GB of memory, the number of QEPs is thus 70 x 260 = 18,200. The problem is then how to optimize such a query, with respect to multiple objectives (response time, monetary cost, etc.).

A. Pareto plan set

Let a **query** q be an information request from databases, presented by a set $\mathcal{Q}$ of tables. A **Query Execution Plan** (QEP) includes an ordered set of operators (select, project, join, etc.). The set of QEPs p of q is denoted by symbol $\mathcal{P}$. The set of operators is denoted by $\mathcal{O}$. A plan p can be divided into two **sub-plans** p_1 and p_2 if p is the result of function $Combine(p_1, p_2, o)$, where $o \in \mathcal{O}$.

The execution cost of a QEP depends on parameters, which values are not known at the optimization time. A vector $\mathbf{x}$ denotes parameters' value and the **parameter space** $\mathcal{X}$ is the set of all possible parameter vectors $\mathbf{x}$. In MOOP, let denote N the set of n cost metrics. We can compare QEPs according to n cost metrics which are processed with respect to the parameter vector $\mathbf{x}$ and cost functions $c^n(p, \mathbf{x})$. Let $\mathbf{C}$ denotes the set of cost function c.

Let $p_1, p_2 \in \mathcal{P}$, p_1 **dominates** p_2 if the cost values according to each cost metrics of plan p_1 is less than or equal to the corresponding values of plan p_2 in all the space of parameter $\mathcal{X}$. That is to say $\mathbf{C}(p_1, \mathcal{X}) \preceq \mathbf{C}(p_2, \mathcal{X}) \mid \forall n \in N, \forall \mathbf{x} \in \mathcal{X} : c^n(p_1, x) \leq c^n(p_2, x)$. The function $Dom(p_1, p_2) \subseteq \mathcal{X}$ yields the parameter space region where p_1 dominates p_2 [13]: $Dom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) \leq c^n(p_2, x)\}$. Assume that in the area $\mathbf{x} \in \mathcal{A}, \mathcal{A} \subseteq \mathcal{X}$, p_1 dominates p_2, $\mathbf{C}(p_1, \mathcal{A}) \preceq \mathbf{C}(p_2, \mathcal{A})$, $Dom(p_1, p_2) = \mathcal{A} \subseteq \mathcal{X}$.

p_1 **strictly dominates** p_2 if all the values for the cost functions of p_1 are less than the corresponding values for p_2 [14]: $StriDom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) < c^n(p_2, x)\}$.

A **Pareto region** of a plan is a space of parameters where there is no alternative plan that has lower cost than itself [15]: $PaReg(p) = \mathcal{X} \setminus (\bigcup_{p^* \in \mathcal{P}} StriDom(p^*, p))$

B. Non-dominated Sorting Genetic Algorithms

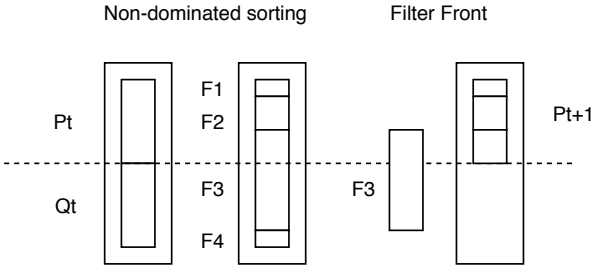

Fig. 1: NSGA-II and NSGA-III procedure [7], [8]

Among EMO approaches, Non-dominated Sorting Genetic Algorithms provide low computational complexity of non-dominated sorting. Initially, NSGAs start with a population, P_0, consisting of N solutions. In query processing in cloud computing, a population represents for a set of candidates QEP. The size of P_0 is smaller than the space of all candidates. Each solution is on a specified rank or non-domination level (any solution in level 1 is not dominated, any solution in level 2 is dominated by one or more solutions in level 1, and so on). At first, the offspring populations Q_0, N solutions, is created by the binary tournament selection, and mutation operators [16]. Secondly, a population $R_0 = P_0 \cup Q_0$ with the size of $2N$ should be divided into subpopulations based on the ordering of Pareto dominance. The appropriate N members from R_0 will

be chosen for the next generation. The non-dominated sorting based on usual domination principle [17] is first used, which classifies R_0 into different non-domination levels ($\mathcal{F}_1$, $\mathcal{F}_2$, and so on). After that, a parent population of next-generation P_1 is selected in R_0 from level 1 to level k so that the size of $P_1 = N$, and so on.

NSGA-II [7] and NSGA-III [8] follow the same process, illustrated by Algorithm 1. Assume that the process is at t^{th} generation. A combined population $R_t = P_t \cup Q_t$ is formed. The population R_t is sorted in level $\mathcal{F}_1, \mathcal{F}_2, ...$ Now solutions that belong to the best non-dominated $\mathcal{F}_1$ are good solutions in R_t. If the size of $\mathcal{F}_1$ is smaller than N, all members of $\mathcal{F}_1$ are selected to P_{t+1}. Thus, solutions in $\mathcal{F}_2$ are chosen next, and so on. This process continues until no more level can be fitted in p_{t+1}. The last level $\mathcal{F}_l$ cannot be filled in P_{t+1}: $\sum_{j=1}^{l} \mid \mathcal{F}_j \mid > N$. The procedure is illustared by Fig. 1.

Algorithm 1 Generation t of NSGA-II and NSGA-III [7], [8]

```
1: function EVALUATION(P_t,N)
2:     S_t = 0, i = 1
3:     Q_t = Recombineation + Mutation(P_t)
4:     R_t = P_t ∪ Q_t
5:     F_1, F_2, ... = Non − diminated − sort(R_t)
6:     while | S_t | ≤ N do
7:         S_t = S_t ∪ F_i
8:         i++
9:     end while
10:    Last front is F_l
11:    if | S_t | = N then
12:        P_{t+1} = S_t
13:        break
14:    else
15:        select N − Σ_{j=1}^{j=l−1} | F_j | solutions in F_l
16:    end if
17:    return P_{t+1}
18: end function
```

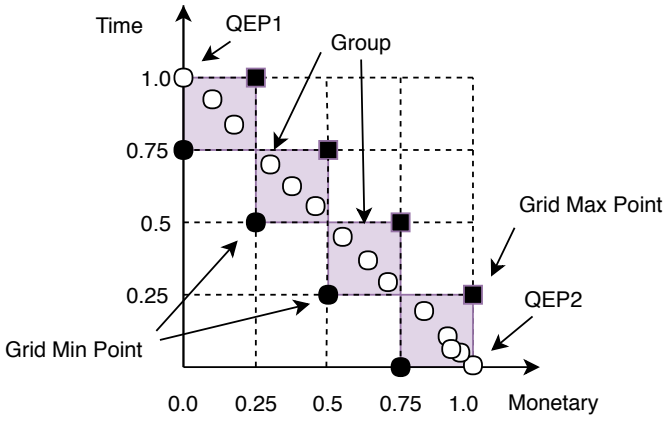

Fig. 2: Grid points and Groups

NSGA-II, NSGA-III and other NSGAs differ in the way to select members in the last level $\mathcal{F}_l$. To keep the diversity, NSGA-II in [7] and SPEA-II [6] use crowding distance among solutions in their selection. K¨oppen and Yoshida [18], [19] stated that NSGA-II procedure is not suitable for multi-objective optimization problems and should replace crowding distance operator for better performance. Hence, when the population has a high-density area, higher than others, NSGA-II will prefer the solution that is located in a lesser crowded region. For example, when the size of the population is 10, NSGA-II will reject four solutions in an area limited by 4 points, near (1.0,0.0), in Fig. 2. In example 1.1, user A and B tend to choose solution (0.0,1.0) or (1.0,0.0), while NSGA-II leads to skipping the particular points closed to (1.0,0.0). For exmaple, the minimum monetary and longest computation time.

On the other hand, MOEA/D [2] decomposed a multiple objectives problem into a various scalar optimization sub-problems. The diversity of solutions depends on the scalar objectives. However, the number of the neighborhood should be defined before running the algorithm, and the authors did not show how to estimate a good neighborhood. The diversity was considered as the selecting solution associated with these different subproblems. The experiments in [8] showed various versions of MOEA/D approaches fail to maintain a good distribution of points

NSGA-III, an Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-Dominated Sorting Approach, Deb et al. [8] using different directions to maintain the diversity of solutions. NSGA-III replaces the crowding distance operator by comparing solutions, each solution being associated to a Reference Point [8], which impacts the execution time to built the Reference Points in each generation. The diversity of NSGA-III is better than others, but the execution time is very high. For instance, with 2 objectives and 2 divisions, 3 reference points will be created, (0.0,1.0), (1.0,0.0), and (0.5,0.5) in Fig. 2. After selecting process, the diversity of population is better than NSGA-II with solutions near 3 reference points. However, NSGA-III should compare all solutions for each point, so the computation time is very high.

In addition, NSGAs often compare all solutions to choose good solutions in $\mathcal{F}_l$. Therefore, when the number of solutions or objectives is significant, the time for calculating and comparing is considerable.

C. Motivation

Using our running example, the traditional approach to build Pareto QEPs set may lead to a space of 18,200 solutions. Indeed, in the worst case of MOOP, all solutions are in Pareto plan set. Using NSGAs, this space would be reduce to the size of the population, N. Nevertheless none of the solution, being NSGA-II or NSGA-III, is satisfactory since there is a tradeoff between complexity and diversity.

III. Nondominated Sorting Genetic Algorithm based on Grid Partitioning

This paper presents our Non-dominated Sorting Algorithm based on the Grid Partitioning (NSGA-G) to improve both

diversity and convergence while having an efficient computation time by reducing the space of selecting good solutions in the truncating process. At t^{th} generation of Non-dominated Sorting Algorithms, P_t presents the parent population with N size and Q_t is offspring population with N members created by P_t. $R_t = P_t \cup Q_t$ is a group which need to be chosen N members for P_{t+1}.

A. NSGA-G

NSGA-G finds the nearest smaller and bigger grid point for each solution. For example, Fig. 2 shows an example of a two-objective problems. If we have the unit of the grid point is 0.25 (the number of grids is 4) and the solution with two objectives values of $[0.35, 0.45]$, the closest smaller point is $[0.25, 0.5]$ and the nearest bigger point is $[0.5, 0.5]$. To avoid calculating multiple objective cost values of all solutions in the population, we divide the space into multiple small groups by Grid Min Point and Grid Max Point, as shown in Fig. 2. Each group has one Grid Min Point, the nearest smaller point, and one Grid Max Point, the nearest bigger point. We only calculate and compare solutions in a group. In this way, in any loop, we do not need to compare solutions among all members in F_l, as F_3 in Fig. 2. Besides, the characteristic of diversity in removing the solution among members is maintained by choosing a group randomly. The next algorithm will show our strategy to select $N - \sum_{j=1}^{l-1} \mathcal{F}_j$ members in $\mathcal{F}_l$.

Algorithm 2 Filter front in NSGA-G

```
1:  function FILTER(F_l, M = N − Σ_{j=1}^{l−1} F_j)
2:      updateIdealPoint()
3:      updateIdealMaxPoint()
4:      translateByIdealPoint()
5:      normalizeByMinMax()
6:      createGroups
7:      while | F_l |> M do
8:          selectRandomGroup()
9:          removeMaxSolutionInGroup()
10:     end while
11:     return F_l
12: end function
```

The functions in line 1 and 2 will determine the new origin coordinates and the maximum objective values of all solutions, respectively. After that, they will be normalized to a range of $0 - 1$. All solutions will be in the different groups, depending on the coefficient of the grid. The most important characteristic of this algorithm is that we select the group randomly like NSGA-III to keep the diversity characteristic and remove the solution among members of that group. This selection helps to avoid comparing and calculating the maximum objectives in all solutions.

B. Quality of NSGA-G

1) Convergence: In terms of convergence, Pareto dominance is a fundamental criterion to compare solutions. Our algorithm keeps the generation process of NSGAs, except the removing solution for the next generation. The convergence characteristic of NSGA-G does not only keep the specification of NSGAs, but also better than original NSGAs. This will be shown in experiments of Generational Distance (GD) [20], Inverted Generational Distance (IGD) [21] in next session.

2) Diversity: Our approach uses Grid Partitioning to guarantee that the solutions are distributed in all solution space. In the problems of N objectives, $N \geq 4$, we assume that the last front should be removed k solutions. In each axis coordinate with the number of grids is n, the maximum of groups in all space of N axis coordinates is n^N, and we chose the maximum of groups in the last front included all non-dominated solutions should be removed is n^{N-1}. Our idea is keeping the diversity characteristic of the genetic algorithm by generating k groups, removing k solutions which have the longest distance to the minimum grid point. Hence, in our algorithm, the number of grids is $n = \lceil k^{1/(N-1)} \rceil$.

3) Computation: In this paper, we study the different reference points based on the grid point to reduce the computing of selecting good solution in the last front. By dividing into small groups, our algorithm selection a good solution in a small group, which has the smaller number of solutions than all of the solutions in the last front.

C. Application to Query Processing in Cloud Computing

In cloud computing, QEPs depend on the number of tables but also on the configuration of virtual machines to be used to execute jobs. Hence, generating the Pareto-optimal front can be computationally expensive and is often infeasible, because of the complexity of the underlying application [3]. Evolutionary algorithms (EAs) are an alternative: they try to look for a good approximation. The set of solution is not too far away from the optimal front. This paper aims to apply NSGA-G to find optimal QEPs in cloud computing, that is to say elastic environments with a the pay-as-you-go model.

Algorithm 3 and 4 show our application of NSGA-G to query processing in cloud computing. A Pareto set of QEPs will be generated after line 5 in Algorithm 3 with the size of QEPs being the size of NSGA's population. Depending on the weighted sum model **W** and users' constraints **B** [22], the best QEP will be chosen by Algorithm 4.

Algorithm 3 Find a query execution plan for a query *Q* in the cloud computing

```
1: function BESTPLAN(Q, W, G, B)
2:     // Find a Pareto set with weight sum model W, configuration G, Constraint B
3:     N ← |Q| //possible logical query plans
4:     M ← |G| //possible VM configuration
5:     P ← NSGA − G(N, M)
6:     //Return best plan in P with weight sum model
7:     return BestInPareto(P, W, B)
8: end function
```

Algorithm 4 Select best plan in $\mathcal{P}$ for weights $\mathbf{W}$ and constraints $\mathbf{B}$

function BESTINPARETO($\mathcal{P}, \mathbf{W}, \mathbf{B}$)
  $P_B \leftarrow p \in \mathcal{P} | \forall n \leq |\mathbf{B}| : c_n(p) \leq B_n$
  if $P_B \neq \emptyset$ **then**
    return $p \in P_B | C(p) = min(WeightSum(P_B, \mathbf{W}))$
  else
    return $p \in \mathcal{P} | C(p) = min(WeightSum(\mathcal{P}, \mathbf{W}))$
  end if
end function

IV. EXPERIMENT

A. Environment

For fair comparison and evaluation, we have used the same parameters such as Simulated binary crossover (30), Polynomial mutation (20), max evaluations (10000), populations (100) for 1. eMOEA [23], 2. NSGA-II, 3. MOEA/D [2] 4. NSGA-III, 5. NSGA-G[1], during their 50 independent runs in solving two types of problems: DTLZ test problems [12] with more than 4 objectives, m, in multiobjective evolutionary algorithms (MOEA) framework [24] in Open JDK Java 1.8. These algorithms use the same population size $N = 100$ and the maximum evaluation $M = 10000$.

B. Results

To estimate the qualities of the different algorithms, we use the Generational Distance (GD) [20], the Inverted Generational Distance (IGD) [21] and the maximum Pareto Front Error [25]. GD measures how far the evolved solution set is from the true Pareto front [26]. IGD is a metric for estimating the quality of approximations to the Pareto front obtained by multi-objective optimization algorithms [27], and could measure both convergence and diversity in a sense. The maximum Pareto Front Error shows the most significant distance between the individual in Pareto front and the solutions in the approximation front [26]. For all metrics, a lower value indicates the better quality.

By dividing the space of solution into multiple partitions and selecting groups randomly, our algorithm has both advantage of diversity and convergence, comparing to other NSGAs. The advantage of NSGA-G is not only shown in the experiments of GD, IGD in Table II, IV, but also is present in the Maximum Pareto Front Error experiment in VI. The convergence and diversity of NSGA-G are often the most or second quality in the tests (Table II and IV).

By comparing solutions in a group, instead of all the space, our algorithm has an advantage of computation time with the high computation problems. Tables III, V, VII show the advantage of NSGA-G in the computation time. It can be seen that NSGA-G is better than other solutions when the number of objectives is large.

[1]https://github.com/dungltr/MOEA

TABLE II: Generational Distance

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	3.675e-02	4.949e+01	1.129e-01	2.494e+00	2.721e-03
DTLZ3	5	1.030e-01	4.418e+00	1.951e-01	7.214e-01	6.342e-03
DTLZ1	6	1.600e-01	9.637e+01	3.138e-01	1.049e+00	3.850e-02
DTLZ3	6	1.306e+01	1.289e+02	5.265e+00	9.577e+00	9.921e-01
DTLZ1	7	1.390e-01	5.283e+01	1.515e-01	4.515e-01	1.542e-02
DTLZ3	7	3.793e-01	3.714e+00	2.251e-02	1.600e-01	2.379e-03
DTLZ1	8	6.817e-01	1.175e+02	2.608e-01	1.949e+00	8.223e-02
DTLZ3	8	1.419e+01	1.667e+02	5.320e+00	1.351e+01	9.146e-01
DTLZ1	9	4.451e-01	4.808e+01	1.101e-01	1.917e+00	1.040e-02
DTLZ3	9	6.843e-02	1.620e+00	5.237e-03	1.280e-01	1.325e-03
DTLZ1	10	3.431e-01	4.340e+01	1.432e-01	2.115e+00	0.000e+00
DTLZ3	10	8.458e-02	1.593e+00	6.763e-03	1.627e-01	1.815e-03

TABLE III: Average compute time (seconds) in Generational Distance experiment

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	5.904e+01	1.063e+02	2.264e+02	4.786e+02	1.261e+02
DTLZ3	5	1.005e+02	1.111e+02	2.358e+02	5.040e+02	1.233e+02
DTLZ1	6	9.024e+01	1.089e+02	2.320e+02	3.509e+02	1.083e+02
DTLZ3	6	1.602e+02	1.243e+02	2.520e+02	3.653e+02	1.209e+02
DTLZ1	7	1.038e+02	1.200e+02	2.839e+02	3.986e+02	1.244e+02
DTLZ3	7	2.946e+02	1.381e+02	2.820e+02	3.565e+02	1.342e+02
DTLZ1	8	1.463e+02	1.313e+02	2.896e+02	4.926e+02	1.249e+02
DTLZ3	8	5.575e+02	1.541e+02	3.458e+02	5.633e+02	1.399e+02
DTLZ1	9	1.573e+02	1.428e+02	3.242e+02	6.823e+02	1.496e+02
DTLZ3	9	8.147e+02	1.988e+02	3.721e+02	8.136e+02	1.640e+02
DTLZ1	10	1.436e+02	1.611e+02	3.745e+02	9.589e+02	1.370e+02
DTLZ3	10	9.151e+02	1.801e+02	3.907e+02	9.805e+02	1.577e+02

TABLE IV: Inverted Generational Distance

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	4.070e-01	8.247e+01	3.434e-01	2.796e+00	3.314e-01
DTLZ3	5	1.656e-01	6.364e+00	3.335e-01	1.383e+00	1.922e-01
DTLZ1	6	7.981e-01	1.786e+02	9.150e-01	3.040e+00	7.034e-01
DTLZ3	6	4.429e+01	4.526e+02	1.164e+01	3.103e+01	8.100e+00
DTLZ1	7	4.188e-01	2.203e+01	3.280e-01	5.024e-01	3.715e-01
DTLZ3	7	9.630e-01	9.286e+00	1.929e-01	3.901e-01	1.667e-01
DTLZ1	8	1.417e+00	2.691e+02	1.023e+00	4.195e+00	9.540e-01
DTLZ3	8	1.023e+02	6.471e+02	1.167e+01	4.194e+01	7.513e+00
DTLZ1	9	4.432e-01	2.396e+01	3.019e-01	6.685e-01	3.147e-01
DTLZ3	9	3.737e-01	3.368e+00	1.381e-01	2.516e-01	1.331e-01
DTLZ1	10	5.912e-01	1.723e+01	3.737e-01	8.963e-01	3.613e-01
DTLZ3	10	6.287e-01	6.049e+00	1.296e-01	5.049e-01	1.521e-01

TABLE V: Average compute time (seconds) in Inverted Generational Distance experiment

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	6.780e+01	9.430e+01	2.292e+02	4.564e+02	9.646e+01
DTLZ3	5	9.976e+01	1.156e+02	2.564e+02	5.036e+02	1.166e+02
DTLZ1	6	7.696e+01	1.078e+02	2.451e+02	3.471e+02	1.178e+02
DTLZ3	6	1.549e+02	1.300e+02	2.527e+02	3.714e+02	1.986e+02
DTLZ1	7	1.021e+02	1.286e+02	2.732e+02	3.271e+02	1.297e+02
DTLZ3	7	3.522e+02	1.942e+02	3.794e+02	3.582e+02	1.523e+02
DTLZ1	8	1.170e+02	1.292e+02	3.222e+02	4.677e+02	1.212e+02
DTLZ3	8	5.333e+02	1.526e+02	3.140e+02	5.190e+02	1.431e+02
DTLZ1	9	1.435e+02	1.812e+02	3.120e+02	7.548e+02	1.544e+02
DTLZ3	9	7.445e+02	2.171e+02	3.533e+02	7.884e+02	1.485e+02
DTLZ1	10	2.104e+02	1.786e+02	3.942e+02	1.532e+03	2.182e+02
DTLZ3	10	1.195e+03	2.526e+02	5.766e+02	1.302e+03	2.131e+02

TABLE VI: Maximum Pareto Front Error

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	7.363e-01	8.969e+02	2.556e+00	2.260e+02	1.024e-01
DTLZ3	5	9.455e+00	1.015e+02	3.692e+00	4.002e+01	1.957e-01
DTLZ1	6	4.699e+00	1.584e+03	8.950e+00	7.488e+01	3.375e-01
DTLZ3	6	5.112e+02	1.862e+03	9.387e+01	4.340e+02	1.244e+01
DTLZ1	7	9.524e+00	1.012e+03	3.074e+00	1.802e+01	1.695e-01
DTLZ3	7	1.458e+01	3.163e+01	2.035e-01	3.116e+00	2.708e-02
DTLZ1	8	3.186e+01	2.041e+03	5.685e+00	2.127e+02	5.532e-01
DTLZ3	8	1.170e+03	2.247e+03	9.867e+01	5.268e+02	1.145e+01
DTLZ1	9	1.111e+01	1.036e+03	2.075e+00	1.496e+02	3.106e-01
DTLZ3	9	1.320e+01	4.065e+01	1.354e-01	8.366e+00	3.195e-02
DTLZ1	10	2.641e+01	1.026e+03	2.793e+00	2.293e+02	0.000e+00
DTLZ3	10	1.492e+01	4.185e+01	1.368e-01	1.079e+01	2.744e-02

TABLE VII: Average compute time (seconds) in Maximum Pareto Front Error experiment

	m	eMOEA	NSGA-II	MOEAD	NSGA-III	NSGA-G
DTLZ1	5	7.454e+01	1.214e+02	2.742e+02	5.796e+02	1.221e+02
DTLZ3	5	1.231e+02	1.437e+02	3.118e+02	6.035e+02	1.286e+02
DTLZ1	6	1.040e+02	1.318e+02	2.848e+02	4.258e+02	1.276e+02
DTLZ3	6	2.166e+02	1.673e+02	3.462e+02	5.014e+02	1.575e+02
DTLZ1	7	1.276e+02	1.638e+02	3.230e+02	4.314e+02	1.424e+02
DTLZ3	7	4.594e+02	1.959e+02	4.188e+02	5.557e+02	1.774e+02
DTLZ1	8	1.637e+02	1.609e+02	3.832e+02	5.952e+02	1.466e+02
DTLZ3	8	5.940e+02	1.963e+02	3.640e+02	6.025e+02	1.453e+02
DTLZ1	9	1.369e+02	1.474e+02	3.148e+02	7.728e+02	1.559e+02
DTLZ3	9	6.596e+02	1.982e+02	3.984e+02	8.069e+02	1.516e+02
DTLZ1	10	1.546e+02	1.540e+02	3.555e+02	9.331e+02	1.400e+02
DTLZ3	10	8.219e+02	1.841e+02	3.601e+02	9.677e+02	1.619e+02

V. Conclusions

In this paper we have introduced a Non-dominated Sorting Genetic Algorithm based on Grid Partitioning improving the diversity and convergence characteristic of Non-dominated Sorting Genetic Algorithms. The proposed algorithm introduced the grid partitioning strategy in selecting, combining parents and offspring solutions to reduce the computation time. Experimental results have shown that the proposed algorithm achieves better computation performance and quality than other algorithms, such as NSGA-II, NSGA-III, and MOEA/D on DTLZ problems with more than four objectives.

In the scope of this paper, our experiments are based on DTLZ problems, which are often used for estimating the quality of Multi-objective Evolutionary Algorithms. NSGA-G will be applied for Multi-objective Query Optimization (MOQO) so as to define data configurations and query processing strategy in an elastic environments with a pay-as-you-go model.

Acknowledgment

The authors would like to thank members of SHAMAN team at Univ Rennes CNRS IRISA and University of Ottawa School of Electrical Engineering and Computer Science for insightful comments.

References

[1] The TPC-H website. [Online]. Available: http://www.tpc.org/tpch/

[2] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712–731, 2007.

[3] L. S. Batista, "Performance Assessment of Multiobjective Evolutionary Algorithms," vol. 7, 2012.

[4] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 602–622, 2014.

[5] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation," in *1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 1, Jul. 1999, pp. 98–105.

[6] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, 2002.

[8] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints," *IEEEXplore*, vol. 18, 2013.

[9] K. Deb, "Multi-objective optimization using evolutionary algorithms: an introduction," *KanGAL Report*, pp. 1–24, 2011.

[10] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization*, Berlin, Heidelberg, 2003, pp. 376–390.

[11] H. Seada, M. Abouhawwash, and K. Deb, "Towards a better balance of diversity and convergence in nsga-iii: First results," in *Evolutionary Multi-Criterion Optimization*. Cham: Springer International Publishing, 2017, pp. 545–559.

[12] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pp. 105–145, 2005.

[13] I. Trummer and C. Koch, "Multi-objective parametric query optimization," *VLDB J.*, vol. 8, 2016.

[14] ——, "A Fast Randomized Algorithm for Multi-Objective Query Optimization," 2016.

[15] A. Hulgeri and S. Sudarshan, "Parametric Query Optimization for Linear and Piecewise Linear Cost Functions," *PVLDB*, pp. 167–178, 2002.

[16] K. Deb and R. B. Agrawal, "Simulated Binary Crossover for Continuous Search Space," *Complex Systems*, vol. 9, pp. 1–34, 1994.

[17] V. Chankong and Y. Haimes, *Multiobjective decision making: theory and methodology*, ser. North-Holland series in system science and engineering. North Holland, 1983.

[18] M. Köppen and K. Yoshida, "Substitute Distance Assignments in NSGA-II for Handling Many-objective Optimization Problems," pp. 727–741, Jan. 2006.

[19] H. Ishibuchi, H. Masuda, and Y. Nojima, "Sensitivity of performance evaluation results by inverted generational distance to reference points," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016, pp. 1107–1114.

[20] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary Computation and Convergence to a Pareto Front," *Late Breaking Papers at the Genetic Programming 1998 Conference*, pp. 221–228, 1998.

[21] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, pp. 163–190, Jun. 2005.

[22] F. Helff and L. Orazio, "Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments," in *EDBT/ICDT Workshops*, 2016.

[23] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, 2002.

[24] (2018) The moea website. [Online]. Available: http://moeaframework.org/

[25] D. A. V. Veldhuizen and D. A. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Evolutionary Computation, Tech. Rep., 1999.

[26] G. Yen and Z. He, "Performance Metrics Ensemble for Multiobjective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, 2013.

[27] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle, "An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Cham: Springer International Publishing, 2017, pp. 31–45.